(12) UK Patent Application (19) GB (11) 2 379 785 (13) A

(43) Date of A Publication 19.03.2003

(21) Application No 0122494.8

(22) Date of Filing 18.09.2001

(51) INT CL[7]
G10L 15/28

(52) UK CL (Edition V)
G4R REX R1F
G3N NGBC2 N286A N286B N286C N381 N391X N403 N407
U1S S1820 S1831 S1834 S1839 S1967 S2105 S2147 S2183 S2204 S2205 S2215

(56) Documents Cited
GB 2368441 A

(58) Field of Search
UK CL (Edition T) G3N NGBC2, G4R REX
INT CL[7] G10L 15/00 15/22 15/26 15/28
Other: Online:WPI, EPODOC, JAPIO

(71) Applicant(s)
20/20 Speech Limited
(Incorporated in the United Kingdom)
Ixworth House, Ixworth Place, LONDON, SW3 3QH, United Kingdom

(72) Inventor(s)
Nicholas John Coleman

(74) Agent and/or Address for Service
Maguire Boss
5 Crown Street, ST IVES, Cambridgeshire, PE27 5EB, United Kingdom

(54) Abstract Title
Speech recognition

(57) A speech recognition system 5 for controlling a functional device 50 on a conveyance has a voice browser 10, 20 which generates a request in response to a user action. The request is conveyed to a document server 70 which is operative to provide a voice mark-up document in reply, the voice mark-up document being used to control the functional device 50.

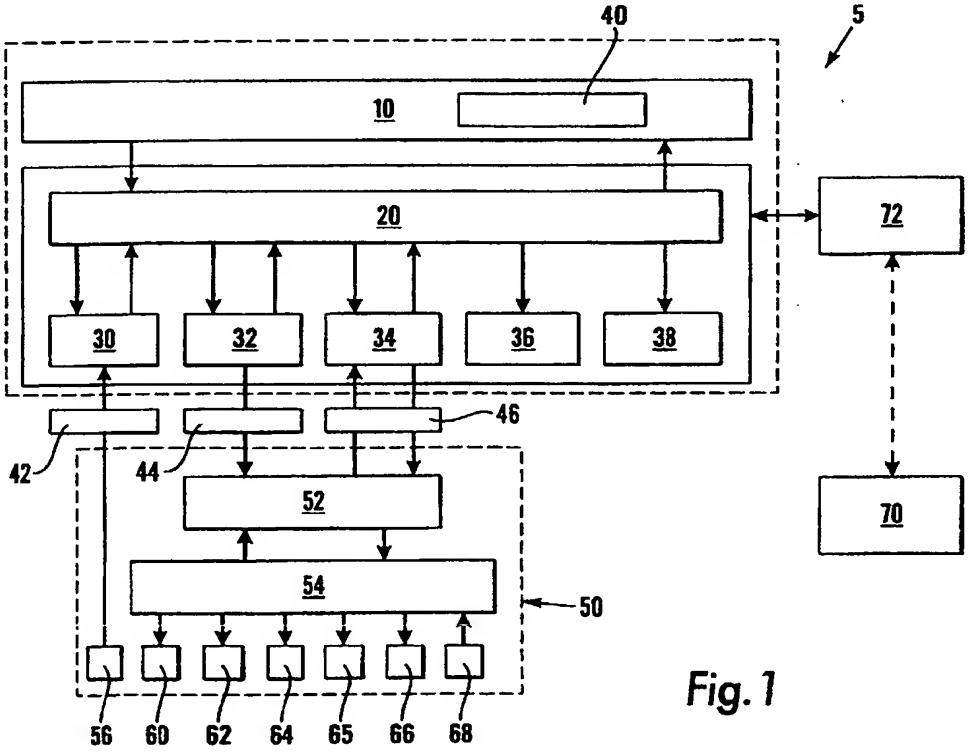

Fig. 1

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

This print takes account of replacement documents submitted after the date of filing to enable the application to comply with the formal requirements of the Patents Rules 1995

BNSDOCID: <GB_____2379785A__I_>

GB 2 379 785 A

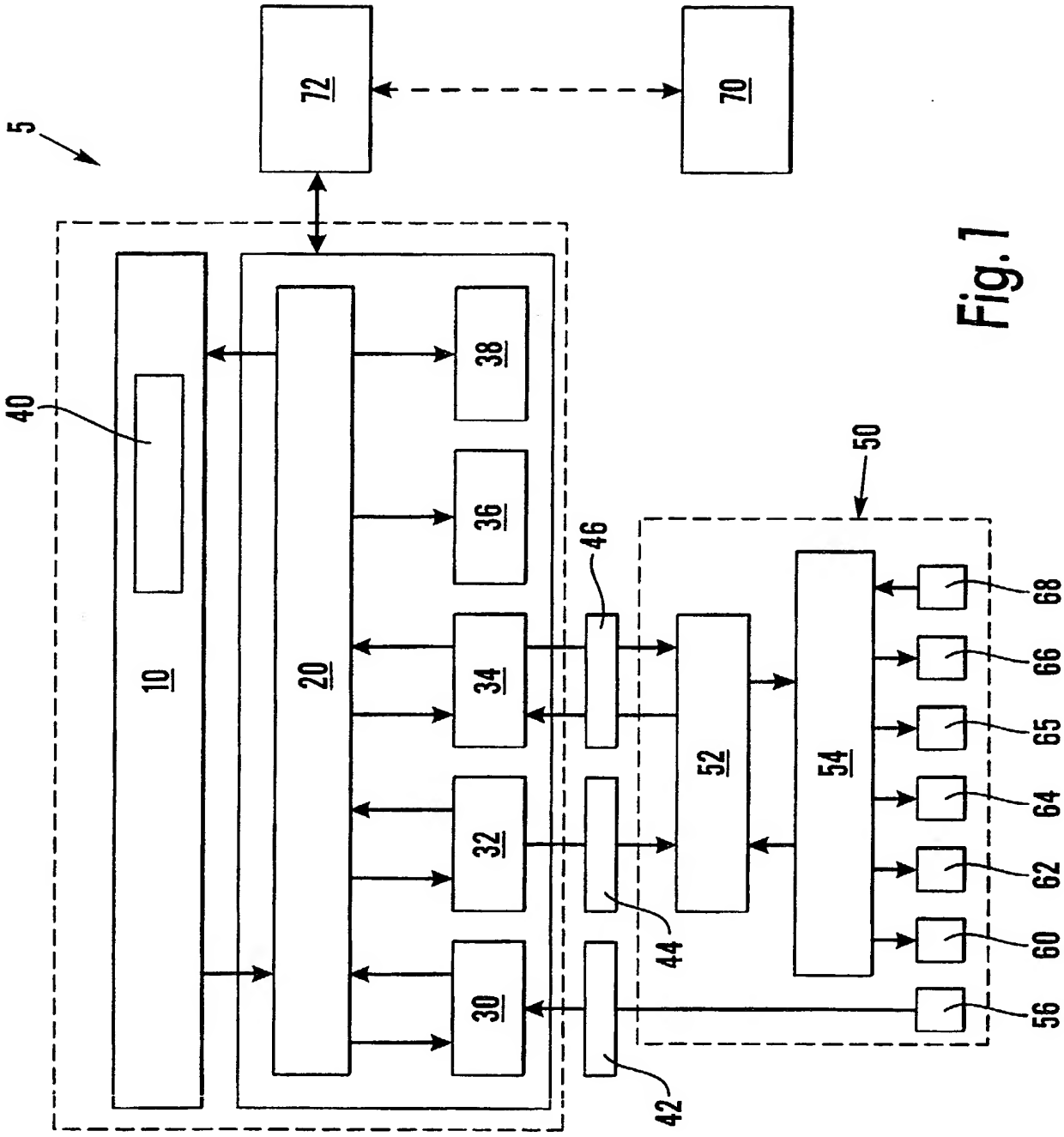

Fig. 1

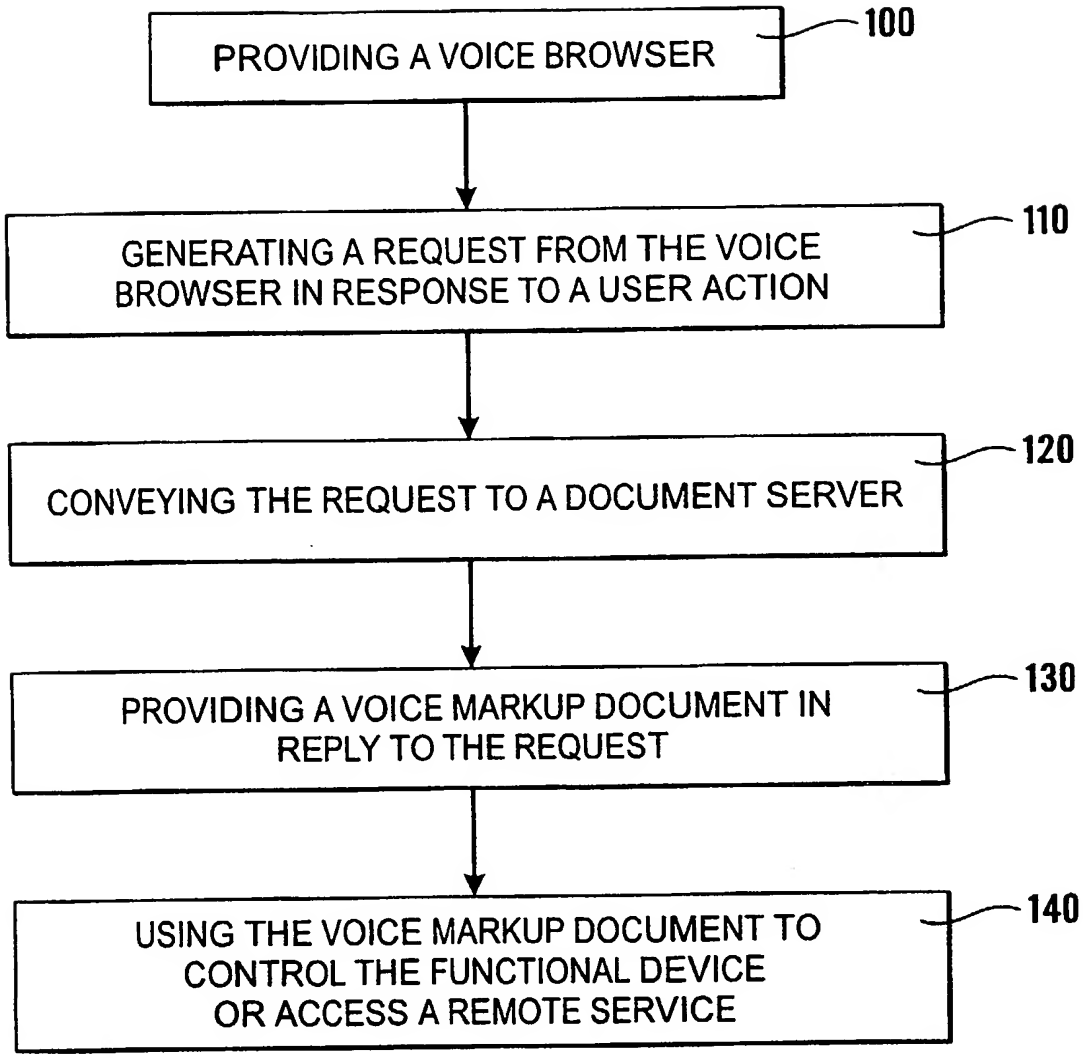

Fig.2

SPEECH RECOGNITION SYSTEM AND METHOD

The present invention relates to a speech recognition system and method, particularly a speech recognition system and method for controlling a functional system on a conveyance, such as an automobile.

Electronic systems have long been implemented as a combination of firmware and hardware modules. Functions of such electronic systems that are not specific to the application in hand can be implemented as one or more hardware modules of a general-purpose design. For example, the one or more hardware modules might include a general-purpose platform, such as a microprocessor board. The use of a general-purpose platform provides for ease of design and for portability across different applications. It is well known that firmware is more readily modified than hardware and thus application specific functions are often implemented in firmware. However, ease of modification can be gained to the detriment of another factor, such as portability or data handling efficiency, particularly where a long used programming language, such as a high-level language, is employed.

The inventors have realised that the use of a mark-up language in an electronic system, particularly a speech recognition system, can provide for advantageous performance, e.g. in respect of portability or data handling efficiency, whilst providing for ease of modification.

Thus, according to a first aspect of the invention there is provided a speech recognition system for controlling a functional device on a conveyance, the speech recognition system having a voice browser, which, in use, is operative to generate a request in response to a user action, wherein the request is conveyed to a document server which is operative to provide a voice mark-up document in reply, the voice mark-up document being used to control the functional device.

Voice mark-up documents are written in a mark-up language, which allows voice dialog behaviour to be specified. Typically, the voice mark-up document presents voice dialog data and voice dialog behaviour data in a form that permits the data elements to be identified and accessed by tags. The voice browser uses the tags to navigate through, download and read voice mark-up documents stored in the document server.

User/document server interactions can also be minimised by specifying multiple interactions in a voice mark-up document. For example, a specific user action might cause the voice browser to return a particular voice mark-up document, which requires the user to select one of several options before control of the functional device is effected.

The voice browser may comprise an implementation platform, which is operative to generate an event in response to a user action and to effect control of the functional device, and a voice mark-up language interpreter, which is operative to translate a generated event into a request and to translate a voice mark-up document into a form executable by the implementation platform.

The partitioning of the voice browser into the implementation platform and voice mark-up language interpreter separates high-level actions, e.g. the generating of a request and the return by the document server of a voice mark-up document, from low level input-output and control actions, e.g. the generating of an event in response to a user action and the control of the functional device. Thus, service portability is promoted across implementation platforms, in that the implementation platform is specific to an application, whereas the voice mark-up language interpreter can be used, or can be readily configured for use with one of many implementation platforms.

The voice mark-up document may be a voice extensible mark-up document, which is a document written in a voice extensible mark-up language, such as VoiceXML. A voice extensible mark-up language permits the creation of structured documents and of data structures, which reflect the content of the structured documents directly and without regard to where the content appears in a document structure. The voice browser can use the data structure as an index to locate and retrieve data from voice extensible mark-up documents. This approach can provide for a more efficient mechanism for data retrieval, particularly where the data is scattered, than navigating directly through voice mark-up documents.

In one embodiment, the document server may be a data store associated with the voice browser. The data store might, for example, be a memory forming part of a hardware implementation of the voice browser.

Alternatively or in addition, the document server may be a data store at a location remote from the voice browser. More specifically, the document server may be a web server. Therefore, in the context of a vehicle the system may provide access to data both on the vehicle and in an external server (for example, by means of a wireless

telecommunications link) and data from these sources may be combined before presentation to a user, perhaps such that the source of the data is not obvious to the user.

According to a second aspect of the invention, there is provided a functional system for installation on a conveyance having a speech recognition system according to the first aspect of the invention. Typically, the functional system includes an automatic speech recogniser.

In a form of this aspect of the invention, the voice browser may be a software module, which is operative to run on a hardware system, such as an embedded microprocessor or microcomputer.

Alternatively or in addition, the functional system may comprise an application specific software module, which is operative to run on a hardware system. An application specific software module might, for example, include one or more of an asynchronous speech recognition input controller, which receives speech from an input device such as microphone, a voice controller, which conducts interaction between the conveyance and the voice browser, and an audio player.

In particular, the functional system may be operative to control one or more functional devices on a conveyance. The functional system may be applied to any functional device that has control of its operation, such as a telephone, an audio system or a navigation system. More specifically, the functional system may be operative to convey data and control information between the voice browser and the functional device via a data path, e.g. an RS-232 connection or a D2B bus, provided on the conveyance.

In an application of this aspect of the invention, the functional system may include an automatic speech recogniser and may be operative to receive at an input device, such as a far field microphone, an utterance spoken by a user, which is provided as an input to the automatic speech recogniser. More specifically, the automatic speech recogniser may be operative to provide a recognition result to the voice browser, which along with the document server may be operative to convey data for control of a functional device, such as a navigation system, on the conveyance.

According to a third aspect of the invention, there is provided a conveyance having a functional system embodying the second aspect of the invention. The conveyance may be a motor land vehicle such as an automobile.

Vehicles with which embodiments of the present invention can be used include conveyances such as land vehicles (e.g. a car), rail vehicles, water-borne vessels or

aircraft.

According to a fourth aspect of the invention there is provided a speech recognition method for controlling a functional device on a conveyance, having the steps of: providing a voice browser; generating a request from the voice browser in response to a user action; conveying the request to a document server; providing a voice mark-up document in reply to the request; and using the voice mark-up document to control the functional device.

According to a fifth aspect of the invention, there is provided a computer program product for causing a computer to carry out the steps of: providing a voice browser; generating a request from the voice browser in response to a user action; causing the request to be conveyed to a document server; and using a voice mark-up document, which is returned by the document server in response to the request, to control a functional device on a conveyance.

In one embodiment, the computer program product may cause the computer to provide a document server. For example, the document server can be provided in the form of a data store in the computer's memory (volatile or non-volatile solid-state storage) or disc drive.

Alternatively or in addition, the computer program product may cause the computer to convey the request to a document server at a location remote from the computer, e.g. a web server.

According to a sixth aspect of the invention, there is provided a method of configuring a speech recognition system, according to the first aspect of the invention, to be operative to control a functional device on a conveyance, the method having the step of configuring a document server, which is associated with the voice browser, to be operable with the functional device.

According to this aspect, the speech recognition system can be configured to operate in an optimal fashion with the functional device in a specific conveyance or kind of conveyance. For example, the document server can be configured by creating one or more voice extensible mark-up documents that are designed for use with a functional device, such as a far field microphone, in an environment provided by a particular marque of automobile. The use of a voice extensible mark-up language, such as VXML, can provide for ease of configuration of a speech recognition system to a particular application by shielding the developer from low-level, i.e. platform specific tasks, and

giving freedom to create and use application specific data structures.

It is to be appreciated that the method according to the fourth to sixth aspects of the invention may include any one or more of the features described above with reference to the preceding aspects of the invention.

Specific embodiments of the present invention will now be described, by way of example, and with reference to the accompanying drawings in which:

Figure 1 is a block diagram representation of a speech recognition and control system being an embodiment of the invention; and

Figure 2 is a flow chart representation of a method embodying the invention.

With reference to Figure 1, the speech recognition and control system 5 comprises a Dialogue Description Language (DDL) interpreter 10 (which constitutes a voice mark-up language interpreter) and an implementation platform 20. The DDL interpreter and implementation platform together constitute a voice browser. The speech recognition and control system also comprises an Aurix module 30 (which constitutes an asynchronous speech recognition input controller), an audio file player 32, a VoiceControl ActiveX control 34 (which constitutes a voice controller), a user interface 36 and a logger 38. Each of the Aurix module 30, the audio file player 32, the VoiceControl ActiveX control 34, the user interface 36 and the logger 38 constitute an application specific software module. In addition, the DDL interpreter includes a VXML parser 40.

The speech recognition and control system 5 can be readily implemented in an embedded microcomputer as is described in more detail below. The document server is implemented in a data store of the rugged PC.

As can be seen from Figure 1, the speech recognition and control system 5 has three interface modules, namely a 'microphone in' module 42, an 'audio out' module 44 and a serial module 46. The three interface modules provide an interface with equipment 50 installed on a car. The 'audio out' module 44 and the serial module 46 provide an interface with an Optolyser module 52, which in turn communicates with a D2B bus 54 on the vehicle. Each of the Optolyser module 52 and the D2B bus 54 constitute a data path provided on the conveyance. The D2B bus is connected to a telephone 60, a navigation system 62, an air conditioning system 64, an in-car entertainment (ICE) system 65, a text display 66 and a press to talk (PTT) switch 68. Each of the telephone 60, the navigation system 62, the air conditioning system 64, the in-car entertainment

(ICE) system, the text display 66 and the press to talk (PTT) switch 68 constitutes a functional device on a conveyance. The 'microphone in' module 42 provides an interface with a microphone 56 installed on the vehicle.

With reference to Figure 1, a web server 70 communicates with the speech recognition and control system 5 by means of a wireless link 72.

According to a mode of operation and with reference to both Figures 1 and 2, a request is generated from the platform and DDL interpreter (which together constitute a voice browser 100) in response to a user action 110, such as operating the PTT switch 68 and speaking into the microphone 56. The request is conveyed to a document server provided in memory in the speech recognition and control system 120. Alternatively, the request is conveyed to the web server 70 (which constitutes a document server), via the wireless link 72, 120. A voice mark-up document is provided in reply to the request 130 and the document is used to control one of the functional devices on the vehicle 140, such as the navigation system 62.

The constitution and operation of the embodiment will now be described in more detail.

The speech recognition and control system 5 is designed for use in cars to allow control of certain functions (for example a telephone, an audio system and a navigation system) using speech recognition.

The embodiment uses Aurix Auto, which is based on the standard Aurix speech recogniser, with the following additions:

1. The speech models used with the recogniser are optimised for use in a car environment and for use with a far field microphone.
2. The speech recognition system is configured to allow the user to use "natural language" rather than a fixed grammar to interact with the car.
3. The system includes a dialog manager that allows the interaction with the car to be specified using an XML based dialog control language.

This embodiment is designed to interface to a road car.

The architecture of the embodiment is shown in Figure 1. The embodiment comprises in-car equipment 50, a rugged Personal Computer (PC), an implementation platform 10 and a DDL interpreter 20.

The in-car equipment provides a convenient interface to the functions of the car that are to be controlled. The data and control connection is a standard (e.g. RS232)

connection and the interface 52 puts the data and control information on the car's internal D2B bus 54.

The functions under control include a telephone 60, a navigation system 62, an in-car entertainment (ICE) system 65 and a text display 66, which is on the instrument cluster and thus visible to the driver. A steering wheel mounted press to talk (PTT) switch 68 is also connected through the bus.

Audio output is through car audio system, with audio input directed to the system through the interface. Thus, mixing and input channel selection is controlled through the car.

The microphone 56 is connected directly to a microphone input of the microcomputer. Internally, the microphone remains connected to the car standard equipment so it can double up as a hands free microphone for the car. No speech control is possible while the phone is active.

The microcomputer is mounted in the boot of a car and connects to the recogniser system as shown in Figure 1. The recogniser system software run on the microcomputer and is designed to run from the time the microcomputer starts up. The user interface 36 does not require any user interaction, although debugging displays can be provided for testing purposes.

The microcomputer runs without a display, mouse or keyboard. A network connection can be used to connect an external computer and to use suitable software to remotely control the software. For example, a battery-powered laptop computer can be used.

Functionality is divided between the platform software code, which handles all input-output functions, and the DDL interpreter 10. This separation of functionality means that the DDL interpreter should not need to change when the code is implemented in another operating system.

The DDL interpreter performs input-output and control functions by calling "objects" implemented in the platform via the VoiceXML OBJECT entity, which is described below.

A list of objects that can be used with the speech recognition system is provided below. Additional objects can be defined to provide functionality required by the system. For example, a HELP object can be defined to simultaneously play audio and to display synchronised messages on the text display.

The Aurix module 30 provides asynchronous speech recognition input and contains methods to control the speech recognition process. The Aurix module is based on Aurix Developer.

The audio file player 32 plays audio files and notifies the platform when playing is complete.

The VoiceControl ActiveX control 34 conducts all interaction with the car. Output functions are represented by methods on the control, and input from the car is received via a standard ActiveX event mechanism. Internally, this control calls a COM EXE server (voice1.exe), which handles the low level communications with the Optolyser.

The user interface 36 shows key status and debugging information. As the system needs to run unattended, the user interface need not be used to start and stop the system. Examples of output features are: list of Aurix output, and status of VoiceControl events (including PTT switch).

The logger object 38 is used to log events to enable tracking and debugging of the system. As a minimum, the logger stores all Aurix output. The Aurix output may include audio where data storage is of a sufficient size.

The DDL interpreter runs the Form Interpretation Algorithm (FIA), which is described below. The DDL interpreter receives input from the platform and invokes functions on the car via objects implemented in the platform, e.g. play audio or send a command to the navigation system via VoiceControl.

The Dialog Description Language (DDL) is based on a subset of the VoiceXML mark-up language, with extensions to facilitate fast implementation in a target environment.

VoiceXML is primarily designed for telephony platforms that have data collection/form filling requirements or that involve menu selection. The present embodiment combines the conventional prescriptive approach with the additional flexibility offered by a more natural language based approach.

VoiceXML also incorporates two levels of including executable content into a document. At a simple level, the language can include IF-THEN-ELSE and GOTO constructs, amongst others, to control flow through the document. VoiceXML also allows the inclusion of ECMA Script (JavaScript) in the same way as the HTML <SCRIPT> element. Only a reduced subset of the first style is implemented for the

purposes of this embodiment.

VoiceXML requires that a speech recogniser allow dynamic loading and activation of grammars, which is a feature not currently available on the Astrec engine. This limitation can be overcome by adding Astrec specific extensions to the language.

VoiceXML provides a Form Interpretation Algorithm (FIA), which is an algorithm for interpreting a document. The Form Interpretation Algorithm of this embodiment is based on that of VoiceXML.

It to be appreciated that the dialogs developed in the DDL of this embodiment are not designed to be fully compliant with VoiceXML and will not run on a VoiceXML platform without modification.

Furthermore, it is to be appreciated that a full VoiceXML implementation can be used if a more powerful speech recognition system is required.

A list of DDL elements used in this embodiment is provided below, along with a definition of each element, examples of use and, where appropriate, a statement of attributes of each element.

- **ASSIGN**

  Assign a variable a value.

  ```
  <assign name="RadioBand" expr="'FM'"/>
  <assign name="RadioSubBand" expr="RadioBand+'1'"/>
  ```

The first of the above examples assigns the string value FM to the variable "RadioBand". The second example illustrates string concatenation. Only string assignment and concatenation are implemented in this embodiment. Variables must first be declared with the VAR element.

Attributes:

name	The name of the variable being assigned to
expr	The new value of the variable.

- **AUDIO**

  Play an audio clip from a file within a prompt.

  <prompt>

  <audio src="c:\sounds\areyousure.wav"/>

  </prompt>

  All audio is held in WAV files - synthesis is not supported in this particular embodiment. Also, it is assumed that files are held on the speech recogniser system and are addressed by a normal path name rather than an Internet URI.

- **CLEAR**

  Clears the value of form items.

  <clear namelist="band frequency preset"/>

  This example resets the values of band, frequency and preset to undefined.

  Attributes:

namelist	The names of the form items to be reset.

- **ELSE**

  See IF.

- **ELSEIF**

  See IF.

- **FIELD**

  A field is an item on a form that can receive a value gathered from the user.

  <field name="Band">

  This simple example declares a field to receive the Band specified by a user. The

speech recogniser will return name/value pairs to facilitate completion of fields. For example, the recogniser may return:

Band=FM

This results in the field being filled.

Attributes:

name Name of the field, which must correspond to one of the defined recogniser names.

- **FILLED**

Specifies an action to perform when the user has filled in some combination of fields. For this embodiment, the actual function required is always returned as the last field, therefore FILLED attributes generally coincide with Function= strings being returned from the speech recogniser.

FILLED elements may be children of FIELD elements or children of FORM elements. When associated with a FORM, it triggers when all of the form items are filled. When on a field, it triggers on the field.

```
<field name="Dial">
    <filled>
        <object name="PhoneDial">
            <param name="Number" expr="MyNumber"/>
        </object>
    </filled>
</field>
```

In this example, object PhoneDial is executed when the Dial field is completed. MyNumber is a variable gathered in another part of the form.

- **FORM**

A form is a collection of field items (for example FIELD elements) and control

elements specifying what to do when the fields are being completed.

```
<var name="MyNumber" expr=""""/>
.
.
<form id="PhoneDialNum">
    <field name="Digit">
        <filled>
            <assign name="MyNumber" expr="MyNumber+Digit"/>
        </filled>
    </field>
    <field name="Dial">
        <filled>
            <object name="PhoneDial">
                <param name="Number" expr="MyNumber"/>
            </object>
            <object name="DisplayText">
                <param    name="Text"
                          expr="__Dialing__"MyNumber""/>
            </object>
            <assign name="MyNumber" expr="""">
        </filled>
    </field>
</form>
```

This example gathers digits into the MyNumber string until the Dial instruction is received. It then calls the PhoneDial object to dial the number and the DisplayText object to put the text into the car instrument cluster before resetting the telephone number variable to an empty string.

- **GOTO**

  GOTO is used to transition to another form or another field in the form

  <goto nextitem="confirm"/>

This syntax example transitions to the field item named "confirm".

```
<goto next="checkexit"/>
```

This syntax example transitions to the form ID "checkexit".

- **IF,ELSE,ELSEIF**

Used for simple conditional logic, allowing tests on string equality only in this embodiment.

```
<if cond="band=='FM'">
    <object name="DisplayText">
        <param    name="Text"
                  expr="__Hi__/__Quality__"/>
    </object>
<else/>
    <object name="DisplayText">
        <param    name="Text"
                  expr="__Lo__/__Quality__"/>
    </object>
</if>
```

- **OBJECT,PARAM**

Within VoiceXML, the OBJECT element is used to define additional platform specific functionality. In this embodiment, OBJECT is used to access the car platform functionality and other implementation specific items. A simple approach is adopted, in which the object name is one of the platform commands and the attributes merely specify parameters to the platform commands.

```
<object name="PhoneDial">
    <param name="Number" expr="MyNumber"/>
</object>

<object name="DisplayText">
```

```
        <param name="Text" expr="__Dialing__/'MyNumber'"/>
    </object>
```

The first example calls the platform command PhoneDial with the specified number. The second example calls the platform function, which displays text strings in the car instrument cluster.

Attributes of OBJECT:

name Name of platform function to access.

Attributes of PARAM

name Name of parameter

expr value of parameter

- **PARAM**

See OBJECT.

- **PROMPT**

Controls the output of audio in a prompt to a user and is limited in this embodiment to playing wave files (see AUDIO). In this embodiment, barge-in is always allowed and timout errors are handled by the implementation (if the user does not respond, for example).

```
<prompt>
    <audio src="c:\sounds\areyousure.wav"/>
</prompt>
```

- **PROPERTY**

Sets a property value that affects platform behaviour, such as parameters for the recogniser and other platform defaults.

```
<property name="timeout" value="5"/>
```

Attributes:

name Name of property

value Value to set

This element can be used to specify the ARGS file.

- **VAR**

Declares a variable. Only string types are supported and all variables are scoped to the whole document.

<var name="MyNumber" expr="''"/>

Attributes:

name Name of variable

expr Initial value of variable

The packaging of this embodiment is described in more detail as follows. The platform must run unattended during operation of the car. Therefore, the system boots and begins running as part of the microcomputer start-up procedure. Although there is not normally the need to see output, a GUI displaying output and other status information can be very useful and thus may be provided.

As a secondary debugging aid, it can be helpful if the same information is output to a file and to a serial port. A device can be attached to the serial port to capture the output in real-time if problems occur. For example, a Psion with a serial cable can be attached to the serial port.

The system is implemented in process, with platform objects implemented as classes within the Platform class. All device and hardware dependencies are encapsulated within the platform and its objects. The DDL Interpreter is device and operating system independent thereby facilitating portability, e.g. from a development system to a target operating system in production versions of the system.

As regards platform implementation, the platform owns an interpreter object, which runs in its own thread. Recognition results and other asynchronous events (timeouts and errors) are returned to the interpreter via a thread safe queuing mechanism.

The interpreter is passed a reference to the owning platform class on initialisation, and the interpreter causes action on the platform by calling methods on this interface.

The platform is solely responsible for collecting user input, controlled by the press to talk switch. No condition should exist which allows the platform to lose spoken input.

Before a user speaks, he/she presses the press to talk switch. This enables recognition. The system stays listening until either a timeout expires or the user completes an utterance.

In this embodiment, the order of initialisation is as follows:

1. Initialise the platform object, passing it the name of the XML file to use for this session. This will be received on the command line when the application is run. This stage comes first, as other initialisation data is held within the XML file.
2. Establish communication with the car. Wait for APPAWAKE event. If not received within n seconds (timeout determined from XML parameter), exit and return major error.
3. Initialise Aurix with SAV filename from PROPERY tags.

All other platform activity is either initiated as calls on the "OBJECTS" by the interpreter or follows one of certain platform events, described as follows.

Platform events occur as:

1. Events from VoiceControl ActiveX control;
2. Timeouts; or
3. Recognition results.

The table below indicates the action taken in each event. Events are received asynchronously by the Platform. All are implemented, handled and logged in this embodiment.

Event	Description	Action
APPAWAKE	This event confirms that communication with the car sub-systems has been successful.	Wait for this before commencing speech recognition. If not received, exit with major error.
APP SLEEP	Sometimes when things go wrong,	If it is, log it and wait for another

Event	Description	Action
	this event will be received.	APPAWAKE. If none comes within a specified time, exit and report fatal error.
ARGUMENTS WRONG	Error in arguments supplied in VoiceControl call.	Log and report as error.
AUDIO OUTPUT OK	Sent when audio output is complete. Need to wait for this before ending the voice session.	
PTT BUT PRESS	Press to talk button has been pressed. A Voice Session will be started, a tone played to the user and the car audio output suppressed.	Stop current audio output. Start timer for "no input" condition. Resume recogniser.
PTT BUT RELEASE	Button has been released.	Useful to know, but should be ignored.
VOICE START	Voice session has started.	Wait for this before playing audio output.
RADIO NOT TUNED		

Like events, all timeouts are logged.

Timeout	Started When	Reset When	Action when received
No Input	PTT button press event received	Recognition result received from Astrec	Add event to Interpreter queue Call VEndVoiceSession Pause Astrec

Recognition results are returned as name/value pairs to the interpreter via a thread safe queue mechanism (the queue is also used to pass back timeouts and events as

appropriate).

As described above, the interpreter causes actions by calling objects implemented in the platform. A custom dispatch type of interface forms the interface between the interpreter and the platform to ensure platform independence.

The Aurix object supports the Aurix immediate and queued commands. This enables features such as syntax switching to be controlled from the dialog in the XML. Reference may be made to the Aurix Developer documentation for further information.

Details of parameters are as follows:

Command	Immediate
String	Command string to send to Astrec.

The car feedback object controls two feedback mechanisms provided in the car, namely audio playback and the display of text on a two-line display on the dashboard.

Details of parameters are as follows:

Audio — String specifying the sequence of audio segments to play. This consists of a comma-separated list of WAV filenames and other numbers and text to be displayed.

In its simplest form, this will be a single WAV file name

expr="prompt1.wav"

or multiple WAV files

expr="prompt1.wav,prompt2.wav".

In this case, the second prompt is played immediately after the first.

Numbers and text such as nametags are treated in a different way. Also, numbers may be spoken in a different way depending on the context. A prefix on the name indicates the type of a number or text string as follows:

P01684585115	Phone Number
N12345	General Number
M1089	Medium wave frequency
F98.4	FM frequency
L200	Long wave frequency
C20.5	Temperature

	TWARWICK Text string
	So a more complex output string can be, for example
	"prompt1.wav,F98.4,prompt2.wav,TWARWICK,prompt3.wav"
Text	Text to display in the in dash display as this audio is spoken.

If the object is called several times, the audio is queued. As each new audio item is reached, the text display is switched to the new text to correspond to the new audio being played.

An implementation technique is to use Windows wave audio functions, as this provides the maximum control.

At any time the platform can request that the player stops output (normally on receipt of a PTT button message). This immediately stops output (taking care to correctly free up the audio buffers).

The car interface forms the primary interface with the vehicle. Each parameter maps to a method in the VoiceControl ActiveX control.

The methods of this embodiment are the prototypes generated by Visual Studio. A list of methods used in this embodiment is provided below. The function of each listed method is self-explanatory.

VOICECONTROL method	Object Parameter
void VAutostore();	
void VPhoneMem(short* num);	
void Vcdcselecttrack(short* Vtrack);	
void VAllByPass();	
void VAudioOff();	
void VCancelGuide();	
void CDCPLAY();	
void VCDCMixAll();	
void VCDCMixOff();	
void VCDCMixTracks();	
void VClimateAuto();	
void VClimateDefrost();	

VOICECONTROL method	Object Parameter
void VClimateOff();	
void VClimateOn();	
void VClimateCycle();	
void VDolbyOff();	
void VDolbyOn();	
void VEndVoiceSession();	
void VGetStationInfo();	
void VHeadUp();	
void Vhome();	
void VNorthUp();	
void VPhoneOff();	
void VPhoneOn();	
void VPhoneRedial();	
void VRadioSeekDown();	
void VRadioSeekUp();	
void VRepeatLast();	
void VReroute();	
void RAvoidFerries();	
void RAvoidTollRoad();	
void VRPrefFerries();	
void VRPrefFastRoad();	
void VRPrefMajorRoad();	
void VRPrefMinorRoad();	
void VRPrefShortRoute();	
void VScreenOff();	
void VScreenOn();	
void VShowAudio();	
void VShowClimate();	
void VShowDestination();	
void VShowJunction();	
void VShowNav();	

VOICECONTROL method	Object Parameter
void VShowPhone();	
void VShowPosition();	
void VShowRoute();	
void VTAOn();	
void VTAOff();	
void VTapere();	
void VTapeFF();	
void VTMusicForward();	
void VTMusicRewind();	
void VTapePlay();	
void VTReverse();	
void VTempHigh();	
void VTempLow();	
void VTIOff();	
void VTIOn();	
void VUninstall();	
void VVoiceGuideOn();	
void VVoiceGuideOff();	
void VRadioOn();	
void VSelectBand(BSTR* vband);	
void VCDCDiskandTrack(short* DISK, short* TRACK);	
void VCDCDiskSelect(short* vdisk);	
void VControlTemp(float* vtemp);	
void Vdisplay(BSTR* vtext);	
void VPhoneDial(BSTR* vnumber);	
void VSelectPreset(short* vpreset);	
void VSetPreset(short* vpreset);	
void VNavZoomIN(short* vlevel);	
void VNavZoomLevel(short* vlevel);	
void VNavZoomOut(short* vlevel);	

VOICECONTROL method Object Parameter

void VRadioSetStation(BSTR* BAND, double* freq);

The DDL Interpreter Implementation will now be described.

The XML parser for the DDL is parsed using a suitable XML parser which may be a proprietary or specifically created parser. This creates a data structure in memory that can be searched and processed with ease.

As regards grammars and syntax checking, standard VoiceXML relies on BNF style grammars that can be loaded and selected dynamically during execution. It should be noted that Astrec prefers the syntax switching mechanism.

In this embodiment, the whole syntax is always active so a user can say anything he/she wishes. However, there are cases when closing down to a smaller syntax will occur, for example Yes/No confirmations. These can be implemented using Astrec syntax switching via an OBJECT entity (the standard way to add non standard extensions to VoiceXML).

As regards the Form Interpretation Algorithm (FIA), operation of the DDL Interpreter component is governed by the FIA. Communication between platform and interpreter is described above.

A suggested approach to processing the input and handling it with the DDL read into a DOM is presented below. It is based on the FIA algorithm in the VoiceXML specification.

```
// Parse DDL into a DOM
// Scan up to the first form in the document
// Initialisation Phase: Initialise all variables and field
// items. Create variables for all field items.
while (true)
{
    if (not came from another form's FIA)
    {
        // Select Phase: choose a form item to visit
        // Find the next field item on the form
        // Collect Phase: execute the form item
        // Queue any prompts associated with this for item
```

```
if (<field> selected)
{
   // Enable Astrec and wait for queued recognition result
   // (utterance) or other event
   // Collect utterance or event
}
else if (<object> selected)
{
   // Execute the object
}
}

// Process Phase: process the resulting utterance or event
if (event)
{
    // Events are timeouts or errors.
    // Error beep
}
else
{
    // We have an utterance
    if (utterance matches another form)
    {
        // Transition to that forms FIA in the process
        // phase.
    }
    else
    {
        // Utterance is for this form
        foreach (slot in the utterance)
        {
            // Copy the value into the field items
            // variable
```

```
                    // Set Just Filled flag on this field
                    // Execute the <filled> actions
                }
            }
        }
    }
```

PROPERTY tags allow control parameters to be held within the XML file. Some are pre-defined as part of the VoiceXML specification, others are specific to this embodiment. The tags are as follows:

Timeout	Time after which a "no input" event is thrown by platform. Time starts when user presses PTT switch and continues until Astrec returns a valid recognition result (ms).
appawaketimeout	Time to wait for APPAWAKE message from VOICECONTROL object (ms).
Savfile	Name of Astrec SAV file to be used during initialization

When a user interacts with the system to request information, the information may be in part obtained from sources within the vehicle, and partly from sources external to the vehicle. The remote source is typically server with which communication can be established using a wireless communication link (e.g. using cellular telephony).

For example, control of vehicle functions, such as audio, telephony, climate control, can be handled using pages that are entirely local to the vehicle; no remote access is required. However, if a user makes a request for other information, such as navigation services, traffic information, news information, interactive services, public transport information, etc., then a request for such information will be forwarded to a remote server.

In such as system, all speech recognition tasks are performed by apparatus on the vehicle, local to the user. At no stage, need a speech signal be transmitted to a remote server for recognition. This has several advantages. First, just one recogniser is needed for all on-board and off-board services. Second, speech recognition is performed on a

speech signal that is obtained from a microphone with little intervening processing. In particular, the speech signal does is not passed over a telecommunication link, with consequent degradation in quality, before recognition is performed.

Similarly, speech synthesis can be handled entirely within the vehicle. This provides speech output to a user that has a consistent quality and voice character. This avoids the disjointed effect that can occur when a speech output is generated by concatenation of output from several different synthesisers. A vehicle manufacturer can thereby exercise an element of branding by providing a consistent voice across a range of vehicles.

A further effect of providing local speech recognition and synthesis is that the quality and reliability of the data link between the vehicle and external data sources need not be as high as would be the case if speech signals were to be transferred over the link for recognition by a server system or reproduction to a user.

It is to be appreciated that the systems, methods and other features described with reference to the embodiment discussed above can be combined in other embodiments of the present invention.

CLAIMS

1. A speech recognition system for controlling a functional device on a conveyance, the speech recognition system having a voice browser, which, in use, is operative to generate a request in response to a user action, wherein the request is conveyed to a document server which is operative to provide a voice mark-up document in reply, the voice mark-up document being used to control the functional device.

2. A speech recognition system as claimed in claim 1, in which the voice browser comprises an implementation platform, which is operative to generate an event in response to a user action and to effect control of the functional device, and a voice mark-up language interpreter, which is operative to translate a generated event into a request and to translate a voice mark-up document into a form executable by the implementation platform.

3. A speech recognition system as claimed in claim 1 or claim 2, in which the voice mark-up document is a voice extensible mark-up document.

4. A speech recognition system as claimed in any one of claims 1 to 3, in which the document server includes a data store associated with the voice browser.

5. A speech recognition system as claimed in claim 4, in which the data store includes a memory forming part of a hardware implementation of the voice browser.

6. A speech recognition system as claimed in any one of the preceding claims, in which the document server includes a data store at a location remote from the voice browser.

7. A speech recognition system as claimed in claim 6, in which the document server includes a web server.

8. A functional system for installation on a conveyance having a speech recognition system according to any one of the preceding claims.

9. A functional system as claimed in claim 8, in which the voice browser is a software module, which is operative to run on a hardware system.

10. A functional system as claimed in claim 8 or claim 9, in which the functional system comprises an application specific software module, which is operative to run on a hardware system.

11. A functional system as claimed in claim 10, in which the application specific software module includes one or more of an asynchronous speech recognition input controller, which receives speech from an input device such as microphone, a voice controller, which conducts interaction between the conveyance and the voice browser, and an audio file player.

12. A functional system as claimed in any one of claims 8 to 11, in which the functional system is operative to control one or more functional devices on a conveyance.

13. A functional system as claimed in claim 12, in which the functional system is operative to convey data and control information between the voice browser and the functional device via a data path provided on the conveyance.

14. A functional system as claimed in any one of claims 8 to 13, in which the functional system includes an automatic speech recogniser and is operative to receive at an input device an utterance spoken by a user, which is provided as an input to the automatic speech recogniser.

15. A functional system as claimed in claim 15, in which the automatic speech recogniser is operative to provide a recognition result to the voice browser, which along with the document server is operative to convey data for control of a functional device on the conveyance.

16. A conveyance having a functional system as claimed in any one of claims 8 to 15.

17. A speech recognition method for controlling a functional device on a conveyance, having the steps of:

providing a voice browser;

generating a request from the voice browser in response to a user action;

conveying the request to a document server;

providing a voice mark-up document in reply to the request; and

using the voice mark-up document to control the functional device.

18. A computer program product for causing a computer to carry out the steps of:

providing a voice browser;

generating a request from the voice browser in response to a user action;

causing the request to be conveyed to a document server; and

using a voice mark-up document, which is returned by the document server in response to the request, to control a functional device on a conveyance.

19. A computer program product as claimed in claim 18, in which the computer program product causes the computer to provide a document server.

20. A computer program product as claimed in claim 18 or claim 19, in which the computer program product causes the computer to convey the request to a document server at a location remote from the computer.

21. A method of configuring a speech recognition system as claimed in any one of claims 1 to 7 to be operative to control a functional device on a conveyance, the method having the step of configuring a document server, which is associated with the voice browser, to be operable with the functional device.

22. A system substantially as described herein with reference to the accompanying drawings.

23. A method substantially as described herein with reference to the accompanying drawings.

INVESTOR IN PEOPLE

Application No: GB 0122494.8 **Examiner:** John Donaldson
Claims searched: 1 to 23 **Date of search:** 20 May 2002

29

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.T): G4R(REX); G3N(NGBC2)

Int Cl (Ed.7): G10L 15/00, 15/22, 15/26, 15/28

Other: Online:WPI, EPODOC, JAPIO

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
A, E	GB 2368441 A (COLES), see abstract	-

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.